TECHNO INTERNATIONAL

BATANGAR

NAME- UPLAB RAJAK

ROLL- 33200118002

SUB - OPERATING SYSTEM. (CS 603)

DEP - C.S.E.

ASSIGNMENT

Q1) What do you mean by Operating System? What are the important functions of an Operating System.

Ans) An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.

The important functions of an operating system.

1. Security
2. Control over system performance.
3. Job accounting.
4. Error detecting aids.
5. Coordination between other software and users
6. Memory Management.
7. Processor Management.
8. Device Management.

Q2) What do you mean by Multitasking?

Ans) Multitasking operating system provides the interface for executing the multiple program task by single user at a same time on the one computer system.

Q3) What do you mean by Multiprogramming?

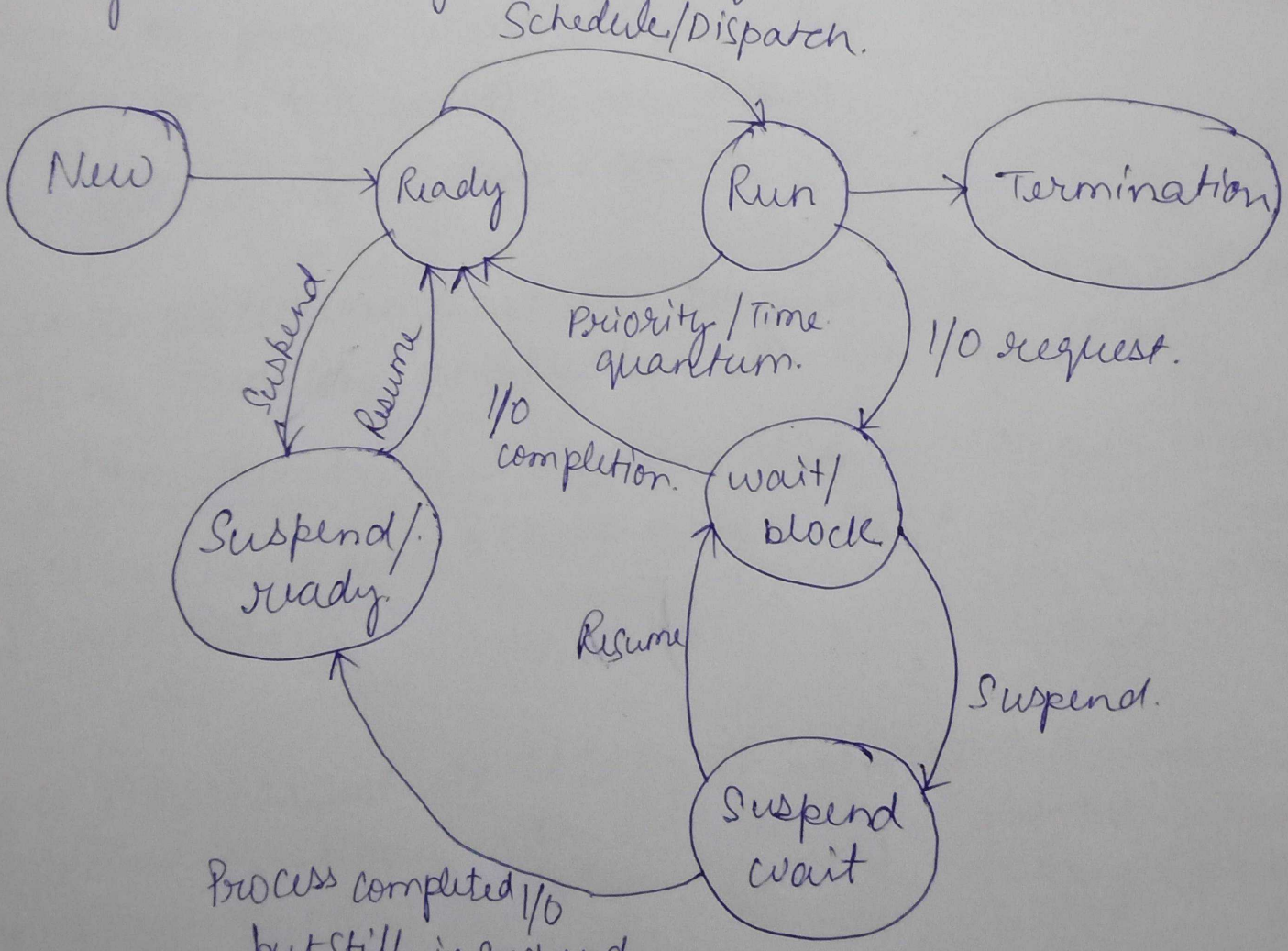
Ans) Multiprogramming is also the ability of an operating system to execute more than one program on a single processor machine..

4) What do you mean by SPOOLING?

Ans) Spooling is an acronym for simultaneous peripheral operations on line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices.

5) What is process? what are the state of process and describe it with suitable picture?

Ans) A process is the instance of a computer program that is being executed by one or many threads.



6) What is Process Control Block (PCB)? (Describe all information it contain).

Ans) Process Control Block (PCB) is a data structure used by computer operating systems to store all the information about a process. It is also known as a process descriptor.

- When a process is created (initialized or installed), the operating system creates a corresponding process control block.
- Information in a process control block is updated during the transition of process states.
- When the process terminates, its PCB is returned to the pool from which new PCBs are drawn.
- Each process has a single PCB.

7) Write the difference between Long Term Scheduler, Short Term Scheduler, and Medium Term Scheduler.

Long term Scheduler	Short Term Scheduler	Medium Term Scheduler
1) Long term is also known as a job scheduler	1) Short term is also known as CPU Scheduler.	1) Medium-term is also called swapping Scheduler.
2) It is either absent or minimal in a time-sharing system.	2) It is significant in the time-sharing order.	2) This Scheduler is an element of Time-sharing systems.
3) Speed is less compared to the short term Scheduler.	3) Speed is the fastest compared to the short term and medium-term scheduler.	3) It offers minimum speed.

8) What do you mean by context switch with proper example?

Ans) A context switch is the process of storing the state of a process or thread, so that it can be restored and resume execution at a later point. This allows multiple processes to share a single central processing unit (CPU), and is an essential feature of a multitasking operating system.

eg:- In the Linux Kernel, context switching involves switching registers.

9) What is Thread? write the Difference between Process and Thread? what are the benefits of Thread?

Ans) A Thread is a path of execution with in a process. A process can contain multiple threads.

Process	Thread.
1) Process means any program is in execution.	1) Thread means segment of a process.
2) Process takes more time to terminate.	2) Thread takes less time to terminate.
3) It takes more time for creation.	3) It takes less time for creation.
4) It also takes more time for context switching	4) It takes less time for context switching.

Benifits of Thread :-

- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient Communication.

10) What is User Level Threads and Kernel Level Threads? write the difference between User Level Threads and Kernel Level Threads.

Ans) USER-LEVEL THREAD.

The user-level threads are implemented by users and the Kernel is not aware of the existence of these threads. It handles them as if they were single-threaded processes. User-level threads are small and much faster than Kernel level threads.

KERNEL-LEVEL THREAD.

Kernel-level threads are handled by the operating system. directly and the thread management is done by the Kernel. The context information for the process as well as the process threads is all managed by the Kernel.

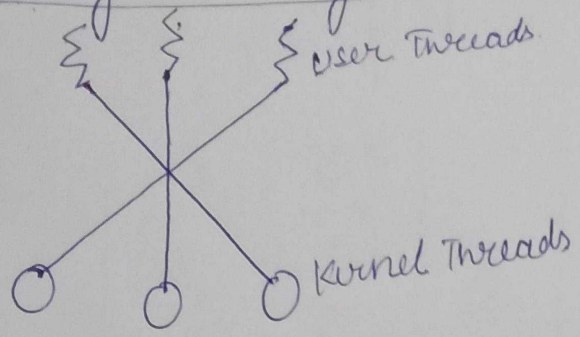
User-Level Threads	Kernel Level Threads.
1) User thread are implemented by users.	1) Kernel threads are implemented by Os.
2) OS doesn't recognized user level threads.	2) Kernel threads are recognized by OS.
3) Implementation of User thread is easy	3) Implementation of Kernel thread is complicated.
4) context switch time is less	4) Context switch time is more.

11) What are the relationships in Multithreading models (between User Level Threads and Kernel Level Threads)? with proper diagram.

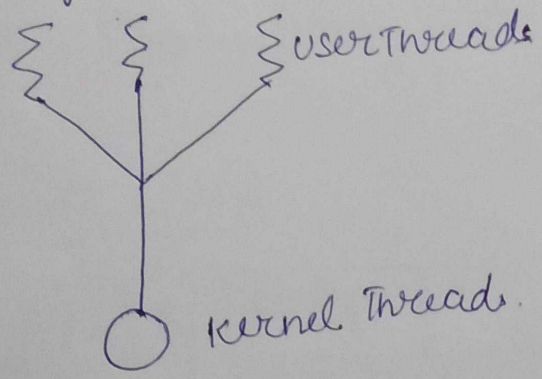
Ans) The relationships in Multithreading models (between User Level Threads and Kernel Level Threads) are.

i) Many to Many Model ii) Many to one model.
iii) One to One Model.

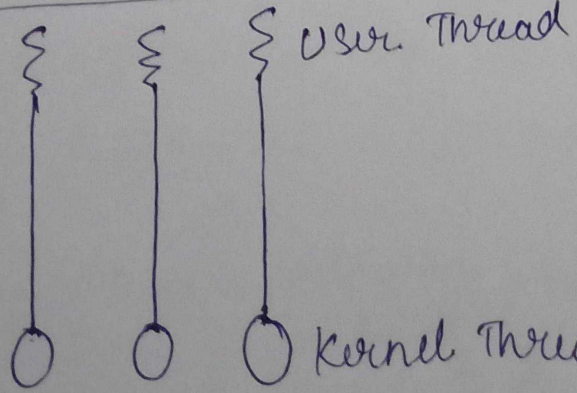
i) Many to Many Model.



ii) Many to One Model



iii) One to One Model



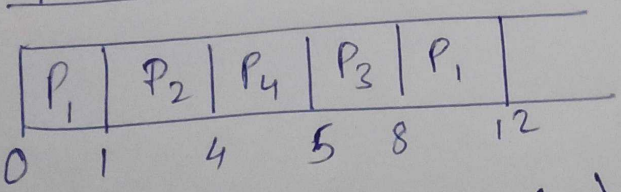
12) Consider the following 4 process with the length of CPU burst time given in millisecond.

Process	A.T	B.T.
P_1	0	5
P_2	1	3
P_3	2	3
P_4	4	1

Ans)

Process	A.T	B.T	C.T	TAT	WT	RT.
P_1	0	5	12	12	7	0
P_2	1	3	4	3	0	0
P_3	2	3	8	6	3	8
P_4	4	1	5	1	0	0.

Gantt Chart.



Avg. T.AT = (12+3+6+1)/4 = 5.5

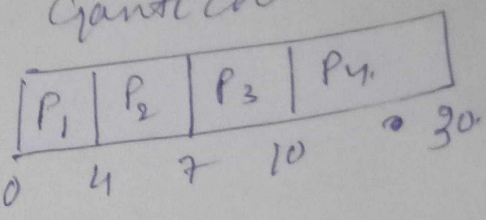
Avg. WT = (7+0+3+0)/4. = 2.5

Avg. Response Time = (0+0+3+0)/4 = 0.75.

13) Consider the following 3 processes with the length of CPU burst time given in milliseconds. Find out the Average waiting Time for RR scheduling. (if the time quantum is 4 ms)

Process	BT	CT	WT
P_1	24	30	6
P_2	3	7	4
P_3	3	10	7

Gantt chart

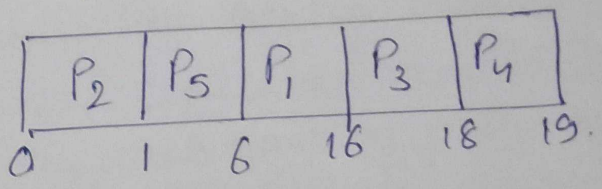


A.W.T = 17/3 = 5.67.

(14) Consider the following processes with the length of CPU burst time given in milliseconds.

Process	B.T	P.
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2.

Gantt chart.



Avg. waiting Time = (1+6+16+18)

15) What is Semaphore? what are the operations on it? Determine the solution of 'Readers-Writers Problem' using Semaphore.

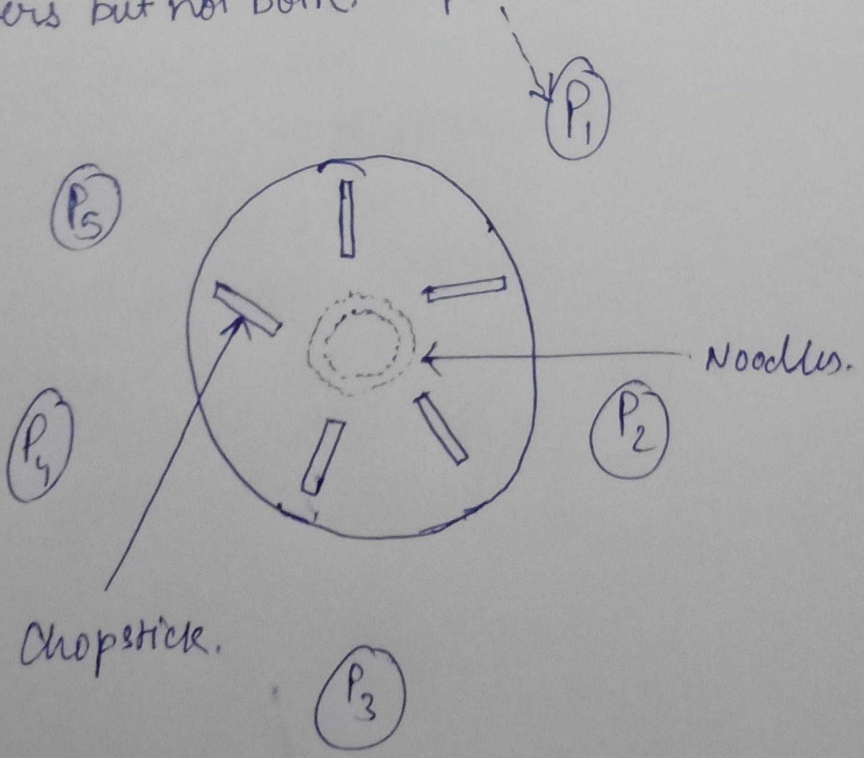
Ans) Semaphore is simply a variable that is non-negative and shared between threads. A semaphore is a signaling mechanism, and a thread that is waiting on a semaphore can be signaled by another thread.

It uses two atomic operations.

1) Wait,

2) Signal for the process synchronization.

The Dining Philosopher Problem – The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philospher. A philosopher may eat if he can pickup the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adajacent followers but not both.



Solution of Readers Writers Problem:

```
Shared data structures;
Semaphore rw_mutex = 1;
Semaphore mutex = 1;
int read_count = 0;
```

Writer Code:

```
while (true) {
    wait (rw_mutex);
    /* writing is performed */
    signal (rw_mutex);
}
```

Reader Code:

```
while (true) {
    wait (mutex);
    read_count ++;
    if (read_count == 1)
        wait (rw_mutex);
    signal (mutex);
    /* reading is performed */
    wait (mutex);
    read_count --;
    If (read_count == 0)
        signal (rw_mutex);
    Signal (mutex);
}
```

16) Determine the solution of 'Dining-Philosopher's Problem' using semaphore.

Ans

```
Semaphore chopstick[5];
while(true){
        wait(chopstick[i]);
        wait(chopstick[(i+1) % 5]);
        /* eat for a while */
        signal(chopstick[i]);
        signal(chopstick[(i+1)%5]);
```

To avoid deadlock, do either one of the following:

1) Allow at most four philosophers.
2) Allow a particular to pick up both chopsticks or none.
3) Use asymetric solution add numbered philosopher picks left chopstick then right and even numbered philosopher picks right chopstick, then left.

(17) Determine the solution of Producer Consumer Problem using semaphore.

Ans)

```
Semaphore empty = 1;
Semaphore full = 1;
Semaphore mutex = 1;

Producer.
        while(true){
            wait(empty);
            wait(mutex);
        /* Producer an item to buffer */
        signal(mutex);
        signal(full);
    }

consumer;(true){
    while(true){
        wait(full);
            wait(mutex);
```

```
/* Consumer an item from buffer */
    signal(mutex);
    signal(empty);
}
```